AF/ JMC

PATENTS
15311-2207
200308268-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re The Application of: Stephen Dickson	)	
Serial No.: 09/428,384	)	Examiner: Ly, A.
Filed: October 28, 1999	)	Art Unit: 2162
For: COMPUTERIZED FILE SYSTEM AND METHOD	)	

Cesari and McKenna, LLP
88 Black Falcon Avenue
Boston, MA 02210
September 14, 2005

CERTIFICATE OF MAILING

I hereby certify that the following paper is being deposited with the United States Postal Service as first-class mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on September 14, 2005.

Melissa L. Altman

_X_ Appeal Brief
_X_ Return Receipt Postcard

_X_ Appeal Brief Transmittal

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200308268-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Stephen Dickson

Confirmation No.: 4583

Application No.: 09/428,384

Examiner: Ly, A.

Filing Date: 10/28/1999

Group Art Unit: 2162

Title: COMPUTERIZED FILE SYSTEM AND METHOD

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on **July 14, 2005**.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( )	one month	$120.00
( )	two months	$450.00
( )	three months	$1020.00
( )	four months	$1590.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of **$500.00**. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: **09/14/2005**

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number_____ on _____

Number of pages: **25**

Typed Name: **Melissa Altman**

Signature: Melissa Altman

Respectfully submitted,

Stephen Dickson

By Michael R. Reinemann

Michael R. Reinemann

Attorney/Agent for Applicant(s)
Reg. No. **38,280**

Date: **09/14/2005**

Telephone No.: **(617)951-2500**

PATENTS
15311-2207
200308268-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re The Application of:
Stephen Dickson

Serial No.: 09/428,384

Filed: October 28, 1999

For: COMPUTERIZED FILE SYSTEM AND METHOD

Examiner: Ly, A.

Art Unit: 2162

Cesari and McKenna, LLP
88 Black Falcon Avenue
Boston, MA 02210
September 14, 2005

CERTIFICATE OF MAILING

I hereby certify that the following paper is being deposited with the United States Postal Service as first-class mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on September 14, 2005.

Melissa Altman

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPEALBRIEF

In response to the Notice of Appeal mailed July 14, 2005, Applicant hereby submits this Appeal Brief.

REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, L.P. of Houston, Texas.

RELATED APPEALS AND INTERFERENCES

Applicant and their legal representatives know of no related appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in the present appeal.

STATUS OF CLAIMS

Claims 1-40 are pending in the case. Claims 1-40 stand finally rejected under 35 U.S.C. §103.

A copy of claims 1-40, in their current form, is attached hereto as an Appendix.

STATUS OF AMENDMENTS

No amendments have been filed since the mailing of the Final Rejection on May 10, 2005.

SUMMARY OF CLAIMED SUBJECT MATTER

The summary is set forth in eleven exemplary embodiments that correspond to independent claims 1, 6, 11, 14, 19, 20, 21, 27, 30, 33 and 36. Discussions about elements and recitations of these claims can be found at least at the cited locations in the specification and drawings.

Independent claim 1 is directed to a computerized file system in which a first process (252), which may reside in a server node (202), maintains a data file (250). A second process (260), which may reside at a client node (204), generates a first message (300) that requests

that the second process be granted a plurality of tokens that are required to modify at least one characteristic of the data file. The first process (252) generates a second message (302) that grants the tokens to the second process, assuming they are available. Specification, p. 13, line 1 to p. 15, line 4, and Figs. 3-5.

Independent claim 6 is directed to a computer node (202) including a first process (252) residing in the node that generates a first message (302) that grants a set of tokens to a second process (260) that requested the grant of the set of tokens, where the set of tokens is required for the second process to modify at one characteristic of a file (250). Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-3.

Independent claim 11 is directed to a computer node (204) including a first process (260) that generates a request to a second process (252) for a set of tokens required to modify at least one characteristic of a file (250). Specification, p. 13, line 1 to p. 15, line 4, and Figs. 3-5.

Independent claim 14 is directed to a network computing system comprising a first node (202) having a data file (250), and a second computer node (204) that issues a first message (300) requesting the grant of a set of tokens for modifying at least one characteristic of the data file, where the first computer node issues a second message (302) to the second computer node that grants the set of tokens if they are available. Specification, p. 11, line 12 to p. 12, line 11, p. 13, line 1 to p. 15, line 4, and Figs. 2-5.

Independent claim 19 is directed to a computer-readable memory (210) containing computer-executable program instructions including a first instruction that maintains a data file (250) in computer storage memory, a second instruction that generates a first message

(300) requesting the grant of a plurality of tokens that are required to modify at least one characteristic of the data file, and a third instruction that generates a second message (302) that grants the tokens, assuming they are available. Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-5.

Independent claim 20 is directed to a computer-readable memory (210) containing computer-executable program instructions including first instructions that generate a first message (302) granting a set of tokens to a requester (260) of the set of tokens, where the set of tokens is required to modify at least one characteristic of a file (250). Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-5.

Independent claim 21 is directed to a computer-readable memory (210) containing computer-executable program instructions including first instructions that generate a request (300) for a grant of a set of tokens required to modify by an issuer (260) of the request at least one characteristic of a file (250). Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-5.

Independent claim 27 is directed to a computerized file system including means for maintaining a data file (processor 212, memory 210, and data file 250 of node 202), means for generating a first message (processor 212 and memory 210 of node 204, and message 300) requesting a grant of a plurality of tokens for modifying at least one characteristic of the file, and means for generating a second message (processor 212 and memory 210 of node 202, and message 302). In response to the first, that grants the tokens if they are available. Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

Independent claim 30 is directed to a computerized method including maintaining a data file (250), generating a first message (300) requesting a grant of a plurality of tokens to modify at least one characteristic of the file, and generating a second message (302) in response to the first, that grants the tokens if they are available. Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

Independent claim 33 is directed to a computerized method including generating a first message (302) that grants a set of tokens that are required to modify at least one characteristic of a file (250). Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

Independent claim 36 is directed to a computerized method including generating a request (300) for a grant of a set of tokens required to modify at least one characteristic of a file (250). Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1, 6, 11, 14, 19, 20, 21, 21, 27, 30, 33 and 36, which otherwise meet all conditions of patentability under Title 35 of the United States Code, are unpatentable under 35 U.S.C. §103 over U.S. Patent No. 5,634,122 issued to Loucks et al. (hereafter "Loucks") in view of U.S. Patent No. 5,506,961 to Carlson et al. ("Carlson"), where the art of record either alone or in combination fails to teach or suggest, among other things, a single message that carries a plurality of tokens that are required to modify a characteristic of a file.

ARGUMENT

Legal Standard

In rejecting claims under 35 U.S.C. §103, the examiner bears the initial burden of presenting a prima facie case of obviousness. See, e.g., In re Rijckaert, 9 F.3d 1531, 1532

(Fed. Cir. 1993). To establish a prima facie case of obviousness, the references, when considered in their entirety, must teach or suggest all of the claimed limitations. If the references fail to teach or suggest any one of the claimed limitations, then the rejection should be reversed. An examiner may not, moreover, resort to speculation, unfounded assumption or hindsight reconstruction to supply deficiencies in the references. In re Warner, 379 F.2d 1011, 1017 (CCPA 1967).

The claims do not stand or fall together. Instead Applicant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-headings as required by 37 C.F.F. §41.37(c)(1)(vii).

Claims 1, 14, 19, 27 and 30

Claim 1 recites as follows:

"A computerized data file system, comprising:"

"a first process that maintains a data file stored in a computer-readable memory", and

"a second process that generates **a first message requesting** that said second process be granted by said first process **a plurality of tokens** required for said second process to modify at least one characteristic of said file stored in said computer-readable memory",

"said first process generating **a second message**, in response to said first message, **that grants said tokens** to said second process if said tokens are available for grant to said second process".

As shown, claim 1 recites, among other things, that there are a **plurality** of tokens being requested by the second process in the **single first message**, and that there are a plurality of tokens being granted by the first process in the single second message.

With regard to the second limitation of claim 1, the final Office Action relies on Loucks as purportedly teaching "a second process that generates a first message requesting that said second process be granted", and on Carlson as purportedly teaching "a plurality of tokens required for said second process to modify at least one characteristic of said file stored in said computer readable memory". The final Office Action further contends that Loucks teaches claim 1's third element, i.e., the "first process generating a second message, in response to the first message, that grants the tokens to the second process if the tokens are available for grant to the second process". As set forth herein, Applicant respectfully disagrees that Loucks or Carlson either alone or in combination teaches or suggests a single message that either requests or grants a plurality of tokens required to modify a file.

Description of the Cited References

Loucks describes a technique for controlling access to shared resources in a distributed computer system using a token manager. See Abstract. In order to access the shared resource, a process (e.g., a client process) must hold a single token associated with the operation. The token represents an authorization for the client process to perform the operation. See Abstract, and Col. 6, lines 8-18. If the client process does not hold the token, it requests the token from a token manager. See Col. 6, lines 12-16. The token manager examines a list of tokens it holds and, depending on the content of the list, may request a new token from a local token manager. Alternatively, it may request that another client process revoke a token, or it may simply grant an available token to the requesting client process. See Col. 6, lines 28-36.

When a token is requested from the local token manager, the local token manager determines if the requested token conflicts with other outstanding tokens. See Col. 7, lines 2-7. If no conflict exists, the local token manager grants the token. Otherwise, if a conflict exists (e.g., the token is granted to another process), the local token manager first revokes the token from the process holding the token, and then grants the token to the requesting client process. See Col. 7, lines 6-12. Specific token requests may cause the revocation of multiple tokens. For example, a request for an exclusive write token on a file causes the token manager to revoke all granted read tokens on that file before granting the exclusive write token to the requesting process. See Col. 7, lines 9-12.

In summary, with Loucks a process requests a single token from a token manager in order to access a shared resource. The token manager, in turn, either grants the token immediately, requests the token from a local token manager, or revokes the token from another process. Moreover if the requested token is not available, depending on the type of the token requested, the token manager may, in turn, revoke several other types of tokens from other processes before granting the requested single token.

Carlson is directed to a peer-to-peer connection authorizer that prevents unauthorized access to a computer system's resources. See Summary of the Invention. With Carlson, when a client device requests access to a server of the computer system, a system authorizer (112) first verifies that the client is a legitimate user of the computer system. If so, the system authorizer issues copies of the same token to both the client and the server. In its request to the server, the client includes a copy of the received token. Col. 6, lines 37-42. The server compares the token contained in the request from the client with the token that the server re-

ceived from the system authorizer. Col. 6, lines 43-45. If they are the same, then the server responds to the client. Col. 6, lines 45-50, and Col. 11, lines 16-44.

If the request by the client involves multiple servers, then the system authorizer sends each of the servers a copy of the token. Col. 4, lines 9-10, and Col. 8, lines 15-18.

Carlson also describes three different token types that may be issued by the system authorizer: a single-use token, a reusable token and a generic token. Col. 8, lines 15-65. A single use token is used once and then destroyed. Col. 8, lines 16-17. A reusable token is issued when the system authorizer determines that continued access to information is required. Col. 8, lines 40-43. A generic token permits free access to information on a particular server device. Col. 8, lines 63-65.

As mentioned above, in rejecting claim 1, the final Office Action contends that Carlson teaches a plurality of tokens requested by a client in order to modify a file. Applicant submits Carlson fails to support such a teaching.

In particular, Col. 4, lines 5-10 of Carlson, which is one of the excerpts relied on by the final Office Action, states as follows:

> Depending on security needs, **the token** may or may not be encrypted. It is also possible that the requisite information resides on more than one server device or that more than one client device requires the information. The system authorizer handles **multiple device** situations by dispatching **multiple tokens** and token copies.

(emphasis added) Applicant submits that this excerpt teaches a system in which each client device requests only one, single token. This is confirmed by Carlson's use of the terms "a token" and "the token" in the above excerpt. Applicant further submits that the reference to multiple tokens by Carlson has to do, not with a single client requesting multiple tokens in a

single message, but with the situation where there are multiple servers. According to Carlson, if there are multiple servers, then multiple tokens will be dispatched; one to each server.

The other parts of Carlson cited by the final Office Action likewise fail to teach or suggest a client request for multiple tokens to modify a file. For example, at Col. 8, lines 6-10, Carlson states that:

> System authorizer mechanism 112 will receive this request in functional block 810. Based on the nature of the request and general system factors, system authorizer mechanism 112 will determine the type of token, the number of tokens sent out, and the distribution of token types 815.

And, at Col. 9 lines 2-5 and 16-25, Carlson states that:

> Once again, system authorizer mechanism 112 could determine that these conditions exist for more than just optical storage devices 340 and send out multiple tokens.
>
> * * *
>
> Once the requested information has been located and the token or tokens of a particular type have been created (and possibly encrypted), system authorizer mechanism 112 bundles it/them into a PTPA message and sends the message(s) to the subject server connection manager(s). FIG. 4 shows the format of the PTPA message in more detail. Field 430 contains the PTPA message type indicator. Field 435 is used within the IPCF protocol to identify the entity for which the authorization is intended.

Applicant agrees that Carlson discusses multiple tokens. However, what these excerpts teach is that, when a client request is to be serviced by multiple servers, Carlson's system authorizer sends to each such server a separate copy of the same token that is also being provided to the client.

In other words, Carlson's reference to multiple tokens has to do, not with a single client requesting multiple tokens, but with the issuance of multiple copies of the same token to different servers needed to satisfy a client request. At most, Carlson's client requests and

receives a single token, which it then forwards to the server as part of its request for information. This is made clear at Col. 4, lines 1-10 where Carlson states as follows:

> If the system authorizer determines that the client device should be allowed to access the information on the subject server device, it then sends **a token** to the server device and a copy of the same token to the client device. Depending on security needs, **the token** may or may not be encrypted. It is also possible that the requisite information resides on more than one server device or that more than one client device requires the information. The system authorizer handles **multiple device** situations by dispatching **multiple tokens** and token copies.

(emphasis added)

Based on the foregoing, Applicant submits that Carlson fails to teach or suggest "a second process" that generates a "first message" requesting a "plurality of tokens" so that the second process may modify at least one characteristic of a file, as contended in the final Office Action's rejection of claim 1.

Applicant also submits that Loucks fails to teach or suggest the third limitation of claim 1, namely, a second message, which is in response to a first message, and that grants a plurality of tokens required to modify a file. The final Office Action cites to Col. 9, lines 18-30 and 40-48, and Col. 11, lines 18-32 of Loucks as purportedly teaching the third limitation recited in claim 1. Applicant respectfully disagree that the cited portions of Loucks provide such a teaching.

The cited excerpts from Col. 9 of Loucks state as follows:

> In addition to the Open tokens 702 and Lock tokens 704, the present invention introduces two new token types to support the above clients: Stash token 706 and Reintegration token 708.
>
> The Stash token is acquired for an entire fileset or for individual files. The stash token is a read-only shared token. It is requested by an MDFS client cache manager (MCM) to indicate this client's intention to disconnect from the server. The client MCM requests the Stash token from the Mobile Protocol Exporter MPX [See

FIG. 5] for an entire fileset. MCM does not need to request stash tokens for the individual files in the fileset. MPX requests the Stash token on the fileset from the local token manager ltkm.

* * *

Since requesting a Stash token indicates that the client is getting ready to disconnect and is probably in desperate need for files in that fileset, an option "as is" is defined to allow the client to get the files. The client can request the Stash token with the "as is" option. If an Open_Exclusive token for the fileset is being held by another client or by a local process on the server, it usually means that there is some repair operation on the fileset. The requesting client will get the stash token with a return code that indicates the fileset structure may be changing. Thus, a request for a Stash token with the "as is" option is always granted.

The cited excerpt from Col. 11 of Loucks states as follows:

If mtkr has not previously acquired the requested token (N output of 906), it sends an Open_Write token request to the MDFS server token manager mtkm (922) and waits for reply (924). If the reply was to grant the token (Y output of 926), the mtkr proceeds to grant the token (916) as described above. If the reply from mtkm was to deny the token request or if time-out was reached before a reply is received (N output of 926), then mtkr proceeds to deny the token request (918) as described above. In both cases, this ends the processing of the current token request operation at mtkr.

FIG. 9b represents the steps executed by the mtkm. They are similar to those explained in description of FIG. 8, with the "Open_Write token" request for fileset y replaced by "Open_Write token" request for file f.

Applicant agrees that Loucks describes multiple types of tokens. See, e.g., Col. 6, lines 9-12. However, the issue is whether Loucks teaches or suggests a process that generates a single message granting a plurality of tokens required to modify a file. Applicant respectfully submits that Loucks fails to provide such a teaching.

First, the above excerpts of Loucks do not teach or suggest the granting of a plurality of tokens with a single message. Instead, the above excerpts from Col. 9 teach the use of a different type of token depending on the particular circumstances, including a Stash token that can be used to obtain a fileset even though the fileset may be undergoing a change. The

above excerpt from Col. 11 teaches a request for a single token (the Open_Write token), which either results in the single Open_Write token being granted or in the request being denied. In sum, the cited excerpts fail to teach or suggest a process that generates a single message granting a plurality of tokens for modifying a file.

A review of other portions of Loucks further demonstrates that a grant is only made of a single token, not a plurality of tokens as recited in claim 1. At Col. 6, lines 28-34, for example, Loucks states as follows:

> When a token requester, mtkr 418, of an MDFS client requests a token from the token manager 420, the token manager examines the list of tokens it holds on behalf of all its clients. Depending on the contents of this list, it may request a new token from the local token manager, request "revoke token" from one or more clients, **or simply grant an available token to the requesting client**.

(emphasis added).

At Col. 10, lines 24-27, Loucks states:

> If an Open_Exclusive token has been previously granted to mtkm (Y output for 824), or if no such token has been previously granted (N output for 822), the ltkm **grants the requested token** (826).

(emphasis added).

At lines 27-39 of Col. 10, Loucks states:

> When a reply is received that indicates the token is relinquished by its current holder (N out of 844), ltkm proceeds **to grant the token** (826).

(emphasis added).

Finally, at Col. 11, lines 13-18, Loucks states:

> If there is no conflict (N output of 908), or if the process holding the conflicting token ends its file system call (Y output of 912), then mtkr proceeds **to grant the requested token** (916). **Granting a token** involves updates to mtkr data structures and sending a "grant token request" to the requesting process.

(emphasis added).

All of the above excerpts demonstrate that Loucks teaches the grant of a single token. There is no teaching or suggestion by Loucks for a single message that somehow grants a plurality of tokens. Because Loucks fails to teach or suggest a single message for granting a plurality of tokens, the rejection of claim 1 should be reversed.

Independent claims 14, 19, 27 and 30 likewise recite a first message that requests "a set" or "a plurality" of tokens, and a second message granting "the set" or "the plurality" of tokens. As set forth above, neither Loucks nor Carlson either alone or in combination teach or suggest such messages. Accordingly, the rejection of claims 14, 19, 27 and 30 should also be reversed.

Claims 6, 20 and 33

Independent claim 6 recites in relevant part:

> "a first message that grants a set of tokens . . . the set of tokens being required for the second process to be able to modify at least one characteristic of a file".

As set forth above, neither Loucks nor Carlson teaches or suggests a single message that grants a plurality of tokens.

This is made clear by reference to the format of Carlson's authorization message 465, which is shown at Fig. 4. As shown, the message 465 only has room to grant a single token, namely the Server token 440. This condition is confirmed by Carlson in his written description at Col. 9, lines 27-28 where he states "Field 440 contains **the actual token** itself" (emphasis added). Thus, by definition, Carlson's authorization message 465 can grant but one token.

Independent claims 20 and 33 likewise recite "a first message that grants a set of tokens". Therefore, the rejection of these claims should be reversed.

Claims 11, 21 and 36

Independent claim 11 recites in relevant part:

> "a first process . . . that generates a request to a second process for a grant **of a set of tokens** required to enable the first process to modify at least one characteristic of a file".

As set forth above in the analysis regarding claim 1, neither Loucks nor Carlson teach or suggest a request for a set, e.g., multiple, tokens. Accordingly, the rejection of claim 11 should be reversed.

Independent claims 21 and 36 similarly recite "a request for grant of a set of tokens". Thus, the rejection of these claims should also be reversed.

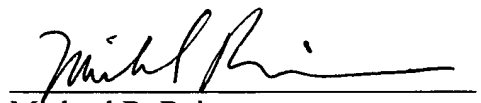
In sum, Applicant submits that the final Office Action fails to establish a prima facie case of obviousness, and that therefore the rejection of claim 1-40 should be reversed.

CONCLUSION

Applicant respectfully submits that the claims are allowable over the art of record.

Accordingly, Applicant request that the rejection of all claims be reversed.

Respectfully submitted,



Michael R. Reinemann
Reg. No. 38,280
Tel. 617-951-3060

Please direct all correspondence to:

IP Administration Legal Department,
M/S 35
Hewlett-Packard Co.
P.O. Box 272400
Fort Collins, CO 80527-2400

CLAIMS APPENDIX
(Claims on Appeal in Appl. Ser. No. 09/428,384)

1. (Previously presented) A computerized data file system, comprising:

a first process that maintains a data file stored in a computer-readable memory; and

a second process that generates a first message requesting that said second process be granted by said first process a plurality of tokens required for said second process to modify at least one characteristic of said file stored in said computer-readable memory;

said first process generating a second message, in response to said first message, that grants said tokens to said second process if said tokens are available for grant to said second process.

2. (Original) A system according to claim 1, wherein:

said first process is resident at a server computer node, and said second process is resident at a client computer node.

3. (Original) A system according to claim 1, wherein:

if any of said tokens are unavailable for grant to said second process as a result of current grant of said tokens to at least one other process, said first process generates a third message revoking the current grant of said tokens to said at least one other process.

4. (Original) A system according to claim 3, wherein:

said at least one other process, in response to said third message, generates a fourth message making said tokens available for grant by said first process.

5. (Original) A system according to claim 3, wherein:

said first process resides in a first computer node;

said second process resides in a second computer node;

said at least one other process resides in at least one other computer node; and

said first computer, second computer, and at least one other computer nodes are networked together and are remote from each other.

6. (Previously presented) A computer node, comprising:

a first process residing in said node that generates a first message that grants a set of tokens, if the set of tokens is available for grant, to a second process that requested grant of the set of tokens, the set of tokens being required for the second process to be able to modify at least one characteristic of a file stored in a computer-readable memory within the computer node.

7. (Previously presented) A node according to claim 6, wherein:

the second process resides in a remote computer node.

8. (Previously presented) A node according to claim 7, wherein:

one of the first and second processes resides in a server computer node and the other of the processes resides in a client computer node.

9. (Original) A node according to claim 6, wherein:

if at least one token in the set of tokens is unavailable for grant because the at least one token is currently granted to a third process, the first process also generates a second message that revokes current grant of the at least one token to the third process prior to generating the first message.

10. (Original) A node according to claim 6, wherein:

the first message is generated by the first process in response to a request for the grant of the set of tokens generated by the second process, the request specifying all tokens required for the second process to be able to modify the at least one characteristic of the file.

11. (Previously presented) A computer node, comprising:

a first process residing in said node that generates a request to a second process for grant of a set of tokens required to enable the first process to modify at least one characteristic of a file residing in a remote computer-readable memory.

12. (Original) A node according to claim 11, wherein:

the second process resides in a second computer node, and the memory is comprised in said second node.

13. (Original) A node according to claim 11, wherein:

the set of tokens comprises all tokens required for the first process to be able to modify the at least one characteristic of the file.

14. (Previously presented) A network computer system, comprising:

a first computer node having a data file stored in a computer-readable memory; and

a second computer node that issues to the first computer node a first message requesting grant of a set of tokens required to carry out a modification of at least one characteristic of said file stored in the first computer node;

the first computer node issuing a second message to the second computer node after receipt of the first message, the second message granting the set of tokens to the first process if the set of tokens is available for grant to the second process.

15. (Previously presented) A system according to claim 14, wherein:

the first computer node is a server node, and the second computer node is a non-server node.

16. (Previously presented) A system according to claim 14, wherein:

the set of tokens comprises all tokens required to carry out the modification of the at least one characteristic of the file.

17. (Previously presented) A system according to claim 14, wherein:

if at least one token in the set of tokens is unavailable for the grant because the at least one token is currently granted, the first computer node waits to issue the first message until after the first computer node receives a third message from a third computer node indicating relinquishment of current grant of the at least one token.

18. (Previously presented) A system according to claim 17, wherein:

the at least one token comprises a plurality of tokens.

19. (Previously presented) Computer-readable memory containing computer-executable program instructions, the instructions comprising:

first instructions which when executed permit a data file to be maintained in a computer storage memory;

second instructions which when executed generate a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file located in said computer storage memory; and

third instructions which when executed generate a second message, in response to said first message, that grants said tokens if said tokens are available for grant to said second process.

20. (Previously presented) Computer-readable memory containing computer-executable program instructions, the instructions comprising:

first instructions which when executed generate a first message that grants a set of tokens, if the set of tokens is available for grant, to a requester of the set of tokens, the set of tokens being required to permit the requester to be able to modify at least one characteristic of a file stored in computer storage memory.

21. (Previously presented) Computer-readable memory containing computer-executable program instructions, the instructions comprising:

first instructions that when executed generate a request for grant of a set of tokens required to enable modification by an issuer of the request of at least one characteristic of a file residing in storage memory.

22. (Previously presented) Computer-readable memory according to Claim 19, further comprising:

further instructions which when executed causes, if any of said tokens are unavailable for grant as a result of current grant of said tokens, generation of a third message revoking the current grant of said tokens.

23. (Previously presented) A computer-readable memory according to claim 22, wherein:

said further instructions, in response to said third message, generate a fourth message making said tokens available for grant.

24. (Previously presented) Computer-readable memory according to claim 20, further comprising:

further instructions which when executed cause, if at least one token in the set of tokens is unavailable for grant because the at least one token is currently granted, generation of a second message that revokes previous grant of the at least one token prior to generating the first message.

25. (Previously presented) Computer-readable memory according to claim 20, wherein:

the first message is generated in response to a request for the grant of the set of tokens generated, the request specifying all tokens required to be able to modify the at least one characteristic of the file.

26. (Previously presented) Computer-readable memory according to claim 21, wherein:

the set of tokens comprises all tokens required to be able to modify the at least one characteristic of the file.

27. (Previously presented) A computerized data file system, comprising:

means for maintaining a data file stored in a computer-readable memory; and

means for generating a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file stored in said computer-readable memory;

means for generating a second message, in response to said first message, that grants said tokens if said tokens are available for grant.

28. (Previously presented) A system according to claim 27, further comprising:

means for generating, if any of said tokens are unavailable for grant as a result of current grant of said tokens, a third message revoking the current grant of said tokens.

29. (Previously presented) A system according to claim 28, further comprising:

means for generating, in response to said third message, a fourth message making said tokens available for grant.

30. (Previously presented) A computerized method for coherently maintaining and modifying a data file, comprising:

maintaining the data file in a computer-readable memory;

generating a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file in said computer-readable memory; and

generating a second message, in response to said first message, that grants said tokens if said tokens are available for grant.

31. (Previously presented) A method according to claim 30, further comprising:

if any of said tokens are unavailable for grant as a result of current grant of said tokens to at least one other process, generating a third message revoking the grant of said tokens.

32. (Previously presented) A method according to claim 31, wherein:

in response to said third message, a fourth message making said tokens available for grant is generated.

33. (Previously presented) A computerized method for use in maintaining coherency of a data file stored in a computer-readable memory, comprising:

generating a first message that grants a set of tokens, if the set of tokens is available for grant, to a requester of the grant of the set of tokens, the set of tokens being required for requester to be able to modify at least one characteristic of the file stored in the computer-readable memory.

34. (Previously presented) A method according to claim 33, wherein:

if at least one token in the set of tokens is unavailable for grant because the at least one token has been currently granted, the method also comprises a second message that revokes current grant of the at least one token prior to generating the first message.

35. (Previously presented) A method according to claim 33, wherein:

the first message is generated in response to a request for the grant of the set of tokens generated by the requester, the request specifying all tokens required for the requester to be able to modify the at least one characteristic of the file.

36. (Previously presented) A computerized method for use in maintaining coherency of a data file stored in a computer-readable memory, comprising:

generating a request for grant of a set of tokens required to enable modification of at least one characteristic of the file stored in the computer-readable memory.

37. (Previously presented) A method according to claim 36, wherein:

the set of tokens comprises all tokens required to be able to modify the at least one characteristic of the file.

38. (Previously presented) The system according to claim 1, wherein:

said second process, in response to receiving said second message, modifies said at least one characteristic of said file stored in said computer-readable memory.

39. (Previously presented) The system according to claim 27, further comprising:

means for modifying said at least one characteristic of said file stored in said computer-readable memory.

40. (Previously presented) The method according to claim 30, further comprising:

modifying said at least one characteristic of said file in said computer-readable memory.

EVIDENCE APPENDIX

None.